Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий
институт
Информационные Системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
Виденин С.А.
подпись инициалы, фамилия
«_____» _____ 2017 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Программный комплекс для расчета эффективности использования альтернативного топлива в котельных

Руководитель _____ к.т.н, доцент Н. В. Молокова
подпись, дата инициалы, фамилия

Выпускник _____ Д. А. Тырышкин
подпись, дата инициалы, фамилия

Нормоконтролер _____ Ю. В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2017

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий
институт
Информационные Системы
кафедра

УТВЕРЖДАЮ
Зав. кафедрой ИС
_____ С.А.Виденин
подпись
«____» _____ 2017 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Тырышкину Дмитрию Александровичу

Группа: КИ13-13Б Направление: 09.03.02 «Информационные системы и технологии»

Тема выпускной квалификационной работы: «Программный комплекс для расчета эффективности использования альтернативного топлива в котельных»

Утверждена приказом по университету № 2517/с от 01.03.2017г.

Руководитель ВКР: Н. В. Молокова, к.т.н, доцент кафедры «Информационные системы» ИКИТ СФУ

Исходные данные для ВКР: технические рекомендации руководителя, учебные пособия, методическая литература.

Перечень разделов ВКР: Введение, анализ предметной области, выбор инструментальных средств и технологий разработки, разработка программного комплекса, заключение, список использованных источников.

Перечень графического материала: Презентация, выполненная в Microsoft Office PowerPoint 2016.

Руководитель ВКР _____ Н. В. Молокова
подпись

Задание принял к исполнению _____ Д. А. Тырышкин
подпись

« ___ » _____ 2017 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Программный комплекс для расчета эффективности использования альтернативного топлива в котельных» содержит 52 страницы текстового документа, 20 рисунков, 14 использованных источников, 1 приложение, 17 формул.

ЭНЕРГЕТИКА, ИНЖЕНЕРНЫЕ РАСЧЕТЫ, АЛЬТЕРНАТИВНОЕ ТОПЛИВО, ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОГРАММНЫЙ КОМПЛЕКС, СУБД, РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА.

Цель работы:

Разработать программное обеспечение для автоматизации инженерных расчетов согласно технологическому процессу перевода котельной на альтернативное топливо.

Основные задачи:

– провести анализ предметной области;
– определить инструментальные средства и технологии разработки;
– разработать программное обеспечение.

Проведено исследование предметной области, разработано программное обеспечение для автоматизации инженерных расчетов согласно технологическому процессу перевода котельной на альтернативное топливо.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Ещё совсем недавно основными инструментами инженера были калькулятор и чертёжная доска. Расчёты занимали немало рабочего времени. В целом, инженерная работа и математические вычисления – итеративный процесс, так как например, большинство инженерных расчётов проводятся в нескольких приближениях, то есть один и тот же алгоритм вычислений повторяется несколько раз, но каждый раз с новыми, уточнёнными данными и инженер вынужден был повторять на калькуляторе вычислительные операции снова и снова. Выполнение чертежей также было нелегким занятием. Так как чертежи выполнялись при помощи карандашей различной твёрдости или туши, то неосторожное движение рукой приводило к появлению на чертежах грязных разводов и смазанных линий. И нередко были случаи, когда при возникновении ошибки приходилось перечерчивать заново весь чертёж. К сожалению, на некоторых предприятиях, работающих «по старинке», такая ситуация существует до сих пор.

В следствие развития научно-технического прогресса на сегодняшний день практически ни одна серьезная разработка в любой отрасли инженерной деятельности не обходится без трудоемких математических расчетов. Обычно, инженеры в своей работе сталкиваются со следующими задачами:

– подготовка научных и технических документов, записанных в привычной для специалистов форме и содержащих текст и формулы;

– проведение и вычисление результатов математических операций с числовыми константами, переменными, размерными физическими величинами;

– проведение серий расчётов с различными значениями начальных условий и других параметров;

– статистические расчёты и анализ данных;

– дифференцирование и интегрирование, аналитическое и численное;

– решение дифференциальных уравнений;

– построение графиков, двумерных и трехмерных.

Конечно, уже далеко не первый год инженеры используют различные средства автоматизации. Автоматизация инженерной работы (расчётов, выполнения чертежей, текстовых документов и др.) позволяет сократить время выполнения инженерного проекта в несколько раз. Также использование компьютеров вместо ручных инженерных расчетов снижает инженерные человеко-часы и стоимость работ.

Для этого требуется оборудовать место работы инженера персональным компьютером и установить на него соответствующее программное обеспечение. Возможности компьютера позволяют использовать его как средство автоматизации инженерной и научной работы.

В данной работе рассматривается проблема автоматизации обработки инженерных расчетов, в данном случае расчетов энергетиков, которая особенно важна на этапе проектирования. Применение информационных технологий для решения такого рода задач позволит существенно расширить возможности специалистов в этой области.

Актуальность данной работы обусловлена сложностью, длительными сроками и малоэффективными результатами проведения работ по автоматизации расчетов эффективности использования альтернативного топлива в котельных.

Объект исследования – обработка и интерпретация в графическом виде инженерных расчетов об эффективности использования альтернативного топлива в котельных.

Предмет исследования – прикладные аспекты технологии программирования к решению задач автоматизации обработки данных.

В связи с вышеизложенным, целью работы является разработка программного комплекса для автоматизации расчетов по оценке эффективности использования различных видов топлива в котельных.

Для достижения указанной цели были поставлены следующие задачи исследования:

– провести анализ предметной области;

– определить инструментальные средства и технологии разработки;

– реализовать программное обеспечение с помощью программных средств.

Данный программный комплекс позволит:

– осуществлять сбор, обработку, хранение информации о котельных установках, нагрузке на них и используемых видах топлива;

– производить вычисления расчётного количества выбросов, полного и удельного продуктов сгорания, подсчитывать экономическую выгоду;

– строить наглядные сравнительные графики по расходу топлива, по выбросам и удельному объему продуктов сгорания при использовании разных видов топлива;

– получать результаты расчетов и расчет экономической выгоды в табличном виде.

1 Анализ предметной области

1.1 Общие сведения

Инженеры сталкиваются с вычислительными задачами в разных областях. Например, одной из наиболее значимых проблем современной экономики и энергетики является проблема обеспечения предприятий и отдельных котельных топливом. Не возобновляемые энергоресурсы истощаются и дорожают, ужесточаются экологические требования, а с развитием промышленности и ростом потребления энергии растет количество и разнообразие отходов. В следствие чего, на мировом рынке возрастает интерес к использованию альтернативных источников энергии – возобновляемых и экологичных, что и обуславливает высокие темпы развития мирового рынка альтернативного топлива. По прогнозу Energy Information Administration (EIA), основанном на данных о мировом потреблении энергии с 1990 по 2012 годы и отображенном на рисунке 3, к 2040 году рост потребления возобновляемых (renewable) энергоресурсов возрастет более чем в 2 раза [1].

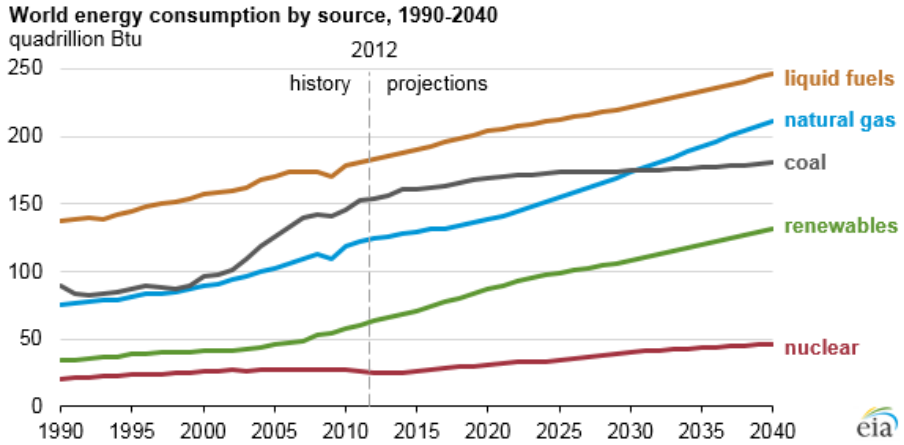


Рисунок 1 – Прогноз EIA мирового потребления энергии до 2040 года

В России топливно-энергетический комплекс (ТЭК) занимает важное место в экономике страны, одним из главных документов комплекса является

энергетическая стратегия, которая должна обновляться не реже одного раза в пять лет. В этой связи Правительством Российской Федерации было принято решение о корректировке Энергетической стратегии России на период до 2030 года с ее пролонгацией до 2035 года. Согласно основным положениям данного проекта, центральной идеей энергетической стратегии-2035 является переход от ресурсно-сырьевого к ресурсно-инновационному развитию ТЭК, а одним из главных стратегических ориентиров является энергетическая эффективность.

На данный момент одним из самых распространенных источников получения энергии является уголь, который содержит много влаги, соединяется легко с кислородом воздуха и при длительном хранении на воздухе сильно выветривается и рассыпается в порошок. Кроме того, он обладает большой склонностью к самовозгоранию.

Использование органической части отходов достаточно успешно решает вопросы дефицита энергии в странах Евросоюза, в США, Японии и других развитых странах. В России на предприятиях, заводах, фабриках также начинают внедряться программы энергосбережения, снижения вредных выбросов в атмосферу, оптимизации использования первичных ресурсов. Например, лесная промышленность в огромных объемах производит отходы, которые переработчики древесины в большинстве случаев используют как топливо для котлов. Они выделяют больше тепла, чем опилки и щепа, не требуют больших складских площадей и при хранении не самовоспламеняются. В научных организациях проводится поиск и исследование новых методов снижения потребляемой энергии и её выработка из нетрадиционных источников энергии, повышения максимальной эффективности её использования, а также поиск способов утилизации отходов производства и бытового пользования.

Самые инновационные технологии, которые будут появляться в будущем, будут относиться к биотопливу третьего поколения. Например, переработка растительного сырья, органических остатков производства, а

также отходов жизнедеятельности сельскохозяйственных животных и т.д., чтобы в результате получить биопродукт – топливо. В настоящее время проводятся исследования в этом направлении. Уже разработано и применяется достаточно широкий спектр видов альтернативного топлива: водород, этанол, метанол, биодизельное топливо и другие.

Например, активное развитие сейчас получают древесные пеллеты – топливные гранулы из древесных опилок. Преимущества пеллет перед другими видами топлива следующие:

– перед газом: высокая пожароопасность и взрывоопасность газа, тяжелая и дорогая процедура согласования, подключения и получения лимитов;

– перед электричеством: высокая стоимость электроэнергии, практическая невозможность подключения нужной мощности;

– перед углем: сжигание угля нельзя автоматизировать, в дымовых газах очень большое содержание серы (до 100 раз больше) и оксидов азота, необходимость утилизировать шлак, достигающих 40% от массы угля, низкий коэффициент полезного действия (КПД) котлов;

– перед дровами: невозможность автоматизировать сжигание дров, нужно много площади для хранения, низкий КПД котлов;

– перед мазутом: высокая стоимость, практическая невозможность применения в малых котлах, необходимость разжижения в холодное время года, до 100 раз больше содержание серы в дымовых газах.

В следствие анализа вышеприведенной информации мною было принято решение изучить технологический процесс перевода котельной на альтернативное топливо, с целью смоделировать расчет эффективности применения различных видов топлива путём разработки соответствующего программного продукта.

Используемый технологический процесс [2], инженерные расчеты которого приведены в приложении А, предполагает решение следующих задач:

а) определение тепловой нагрузки на отопление и горячее теплоснабжение поселка;

б) определение расхода топлива, воздуха, расчет выбросов вредных веществ, объема продуктов сгорания;

в) расчет вспомогательного оборудования котельной;

г) экономическое обоснование предлагаемой замены.

В используемом технологическом процессе рассчитаны значения, необходимые для определения тепловой нагрузки условного поселка, имеются сведения о котельной установке и видах топлива.

Приводятся расчеты расхода топлива, объема теоретического количества воздуха и продуктов сгорания, при сжигании бурого угля (использовался уголь Ирша-Бородинского месторождения) и древесных пеллет.

Также произведен подсчет расчетного количества выбросов летучей золы, оксидов серы и оксидов азота.

Используемый в работе технологический процесс также ссылается на расчеты технико-экономических показателей котельных [3].

В данной работе решается задача автоматизации части технологического процесса, касающейся пункта б).

1.2 Обзор схожих разработок

Прямых аналогов разработанному программному комплексу найдено не было.

Инженерные вычислительные инструменты широко используются в различных инженерных специальностях. Они варьируются от простых шаблонов расчета электронных таблиц, до сложных систем анализа конечных элементов для моделирования напряжений или потоков жидкости. В целом,

для автоматизации математических расчетов используются компьютерные программы узкого специального назначения или универсальные программные средства. В научной и инженерной работе встречается довольно широкий спектр задач ограниченной сложности, для решения которых можно использовать универсальные средства.

К наиболее известным универсальным программным средствам, пригодным для задач, приведенных в подразделе 1.1 относятся такие программы, как:

MathCad – программное обеспечение, позволяющее динамически обрабатывать данные в числовом и аналитическом виде. Mathcad, представленный в 1986 году впервые представил живое редактирование набора математических обозначений в сочетании с его автоматическими вычислениями. Он сочетает в себе возможности проведения расчётов и создания научных и технических документов. Достоинством MathCad является то, описание решения математических задач дается в виде математических формул и знаков, привычных для человека, такой же вид имеют и результаты вычислений.

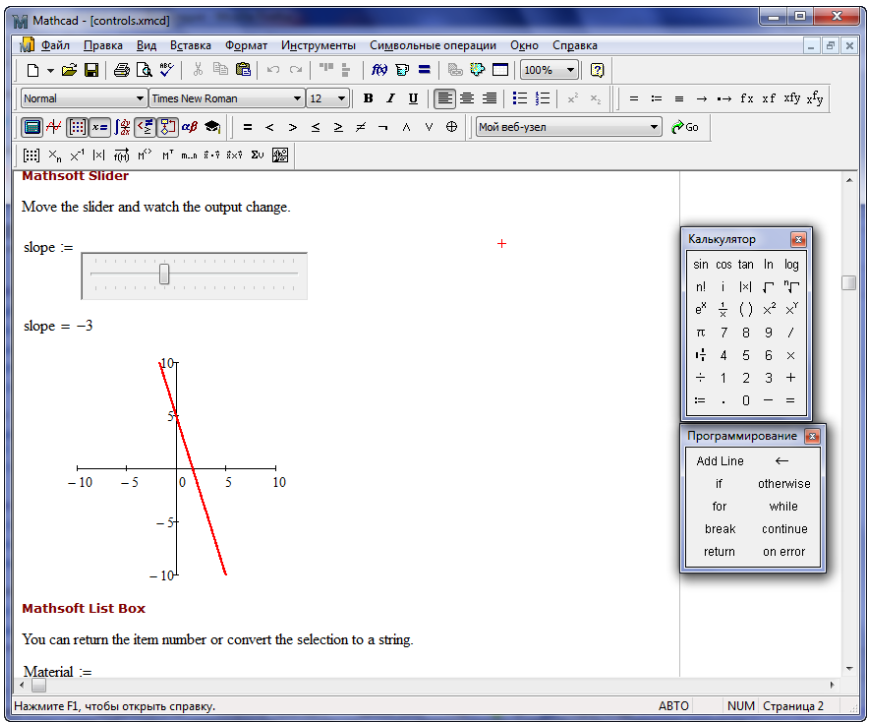


Рисунок 2 - Окно программы MathCad

MathLab – язык программирования и программное обеспечение для научных и инженерных расчетов, построенного на основе интерпретатора этого языка. Достаточно богатый комплекс реализаций современных численных методов компьютерной математики.

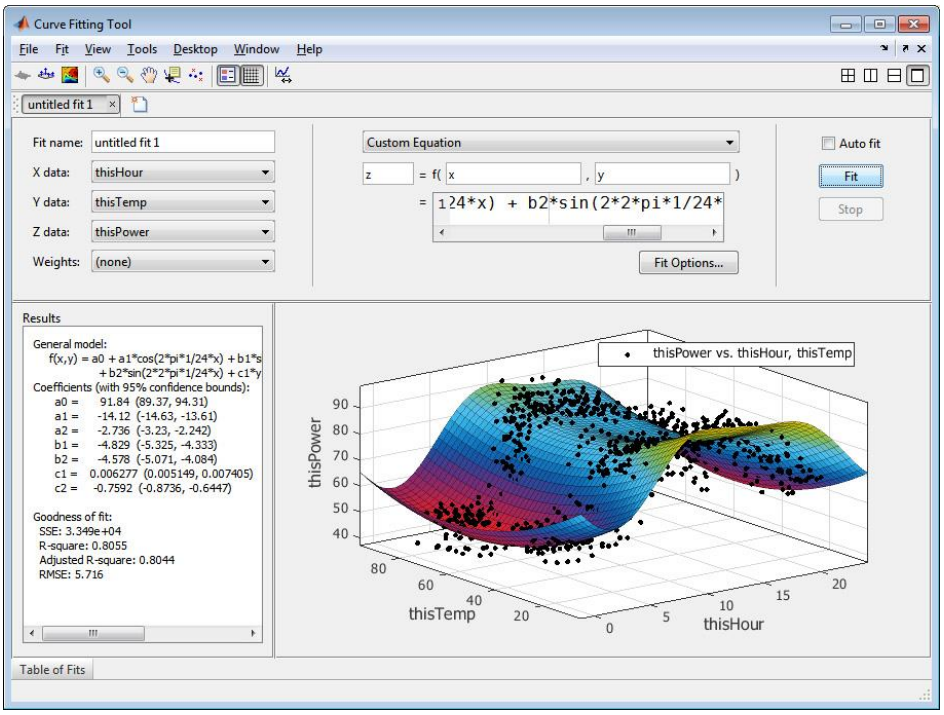


Рисунок 3 - Окно программы MatLab

Microsoft Excel - программное обеспечение, разработанное и изготовленное корпорацией Microsoft, которое позволяет пользователям организовывать, форматировать и вычислять данные по формулам, используя систему электронных таблиц, разбитую по строкам и столбцам. С помощью Excel можно выполнять сложные вычисления с большими массивами чисел и строить диаграммы [4].

Спектр подобных компьютерных средств автоматизации математических расчетов достаточно широк: современная компьютерная математика предлагает целый набор интегрированных программных систем и пакетов программ для автоматизации математических расчетов, таких как

Eureka, Gauss, TK Solver!, Derive, Mathematica, Maple и др., различные CAD/CAE средства.

Недостатками большинства универсальных средств являются дороговизна их лицензионных версий и их системные требования, для комфортной работы таких программ, зачастую, требуется иметь производительный процессор, большой объем ОЗУ и памяти для их установки. Однако, сейчас набирают популярность и «облачные» версии подобных программ. Например – Wolfram Alpha, от создателей Mathematica.

Но зачастую, для решения сложных, узкоспециализированных расчётных задач, либо тех, для которых не подходят универсальные средства, используют программы, написанные специально. И потенциальным пользователям программы, которых слабо интересует изучение языка программирования либо перегруженного интерфейса универсального гораздо удобнее использовать специально написанные средства.

Данный программный комплекс крайне затруднительно было бы выполнить с использованием универсальных средств, так как его реализация требует использования базы данных, также разработанный комплекс имеет возможности к расширению и добавлению новых модулей к нему. Например, можно выполнять интеллектуальный анализ данных.

С учетом вышеперечисленного и стала задача написания программного комплекса для автоматизации расчетов энергетиков согласно техпроцессу перевода котельной на альтернативное топливо.

1.3 IDEF0 диаграмма

Схема бизнес-процесса описывается на основе нотации описания бизнес процессов IDEF0**,** основанной на методологии SADT**.**

SADT (Structured Analysis and Design Technique) - технология структурного анализа и проектирования. Публично появилась на рынке в 1975 г и получила очень широкое распространение в мире.

Графическая диаграмма – главный компонент IDEF0-модели, содержащий блоки, стрелки, соединения блоков и стрелок и ассоциированные с ними отношения.

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Поскольку единственный блок представляет весь объект, его имя – общее для всего проекта. Это же справедливо и для всех стрелок диаграммы. Контекстная диаграмма определяет внешние для системы объекты, которые взаимодействуют с ней, но ничего не отображает о внутренней структуре или поведении системы. Контекстная диаграмма находится на верхнем уровне иерархии. Наиболее важные свойства объекта обычно выявляются на верхних уровнях иерархии; по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции, эти свойства уточняются. На рисунке 4 представлена контекстная диаграмма бизнес-процесса.

Входными данными являются:

– данные о котельной установке;
– нагрузка на котельную;
– константы;
– данные о составе топлива.

Управляющими:

– технологический процесс;
– технико-экономические расчеты;

Ресурсы, выполняющие работу (механизмы):

– пользователь.

Результатом на выходе являются:

– гистограммы расчетного количества выбросов, расхода топлива;
– сравнительные таблицы.

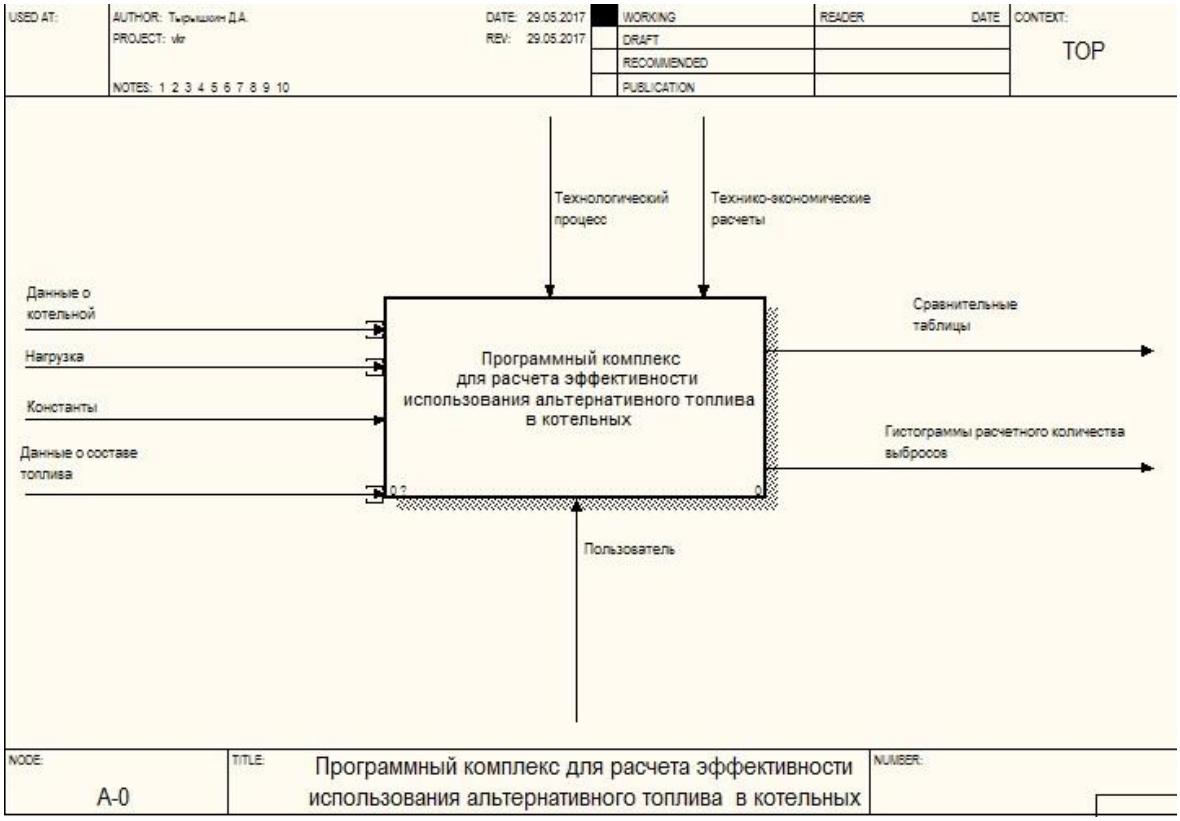


Рисунок 4 – Контекстная диаграмма бизнес-процесса

В состав комплекса входят подмодули:

– модуль управления данными;

– модуль обработки данных;

– модуль отображения данных.

Декомпозиция программного комплекса на модули представлена на рисунке 5.

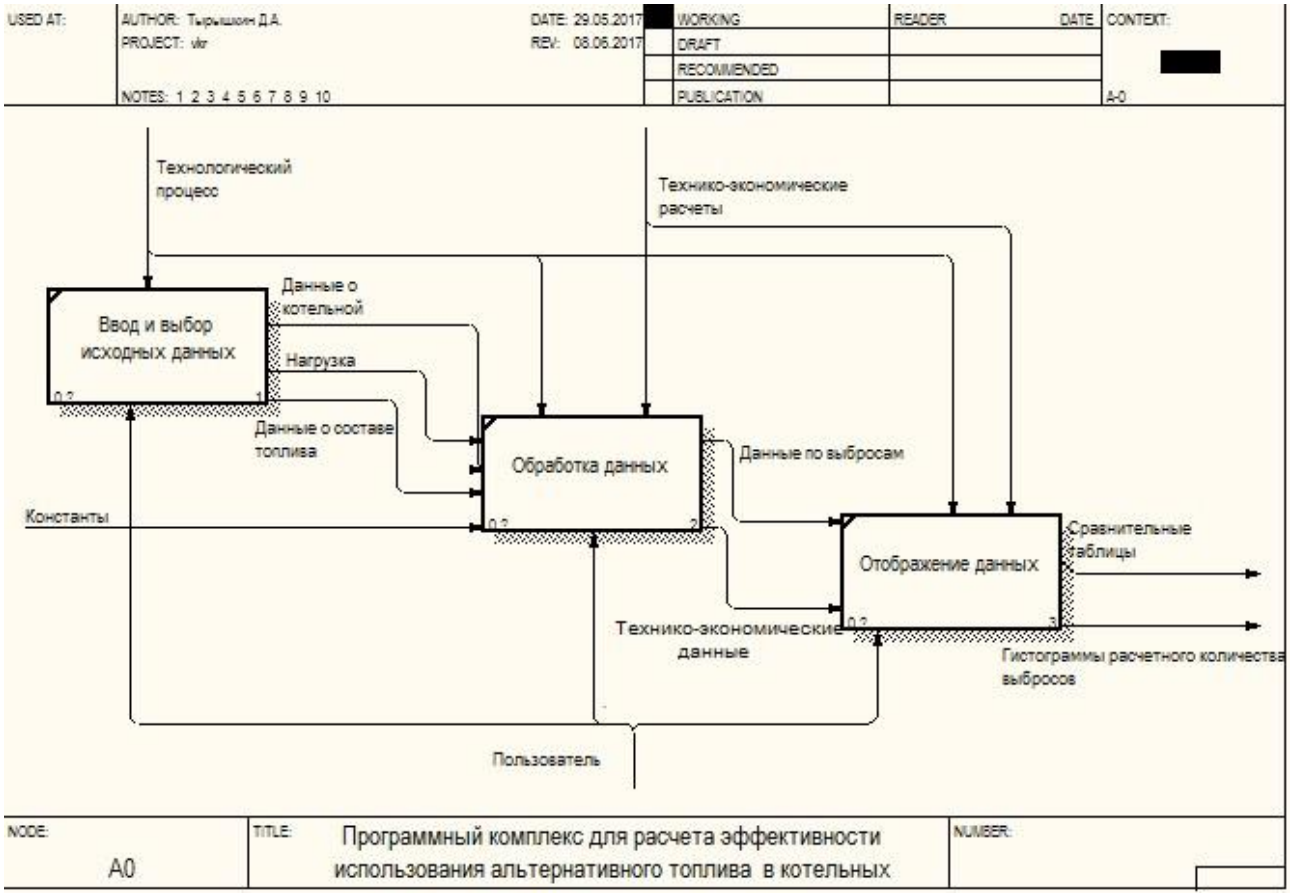


Рисунок 5 – Диаграмма декомпозиции программного комплекса на модули

1.4 Диаграмма вариантов использования

Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).

Диаграмма вариантов использования (use case diagram) - это поведенческий тип диаграммы UML, которая часто используется для анализа различных систем. Диаграммы этого типа описывают статическое представление вариантов использования системы и позволяют визуализировать разные типы ролей в системе и их взаимодействие с системой. Особенно важны для организации и моделирования поведения системы [5].

На диаграмме вариантов использования, представленной на рисунке 6, описана модель разрабатываемого программного комплекса на

концептуальном уровне. На данной диаграмме описаны отношения между пользователем и вариантами использования системы, а именно доступ пользователя к компонентам системы.

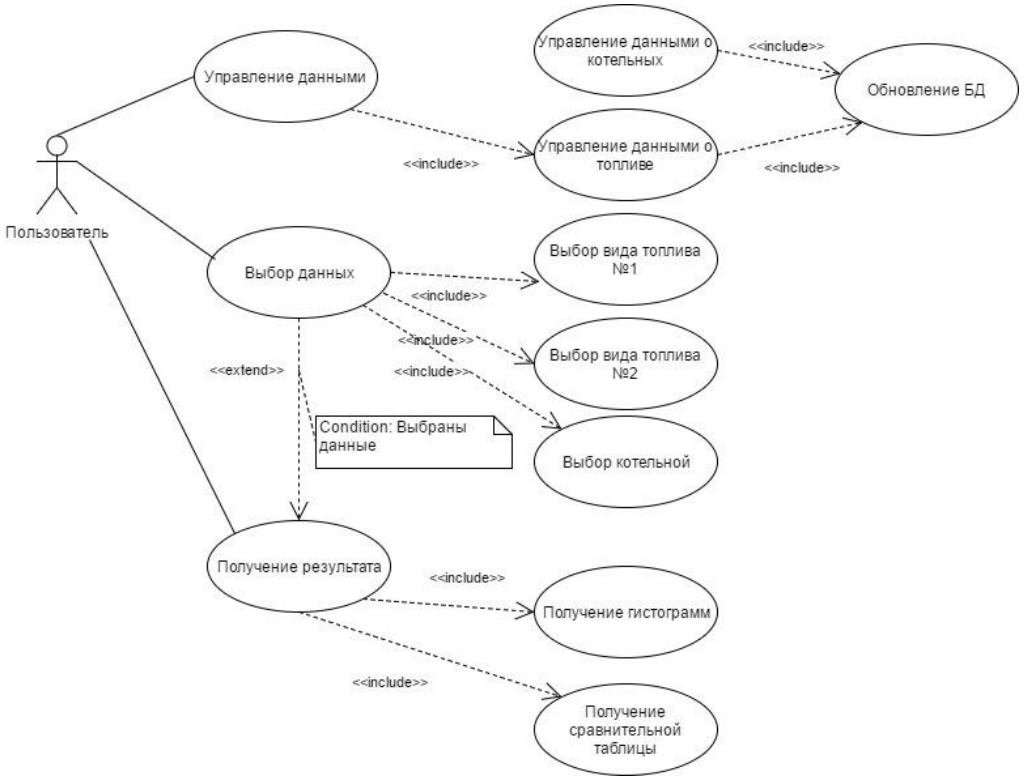


Рисунок 6 – Диаграмма вариантов использования

1.5 Определение требований к проектируемому комплексу

Требования к программному комплексу были определены согласно разделу «Требования к системе» ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы», [6].

Основным функциональным назначением программного комплекса является организация автоматического вычисления количества выбросов (двуокиси серы, двуокиси азота, золы), удельного и полного объема продуктов сгорания, используя в качестве способа представления информации графики и таблицы.

Комплекс должен обеспечить:

– сбор, обработку, хранение информации об используемых котельных установках и видах топлива;

– вычисление расчётного количества выбросов;

– генерацию гистограмм и таблиц на основе выполненных вычислений.

Характеристика объекта автоматизации:

Объектом автоматизации служит набор данных показателей котельной с предварительно рассчитанной на неё нагрузкой и данные о составе используемых видов топлива.

Программный комплекс будет реализован в виде WPF Desktop-приложения и предназначаться для эксплуатации на платформе .NET

Требования к системе в целом:

Требования к структуре и функционированию:

Структура программного комплекса должна состоять из трех подмодулей:

– модуля управления данными;

– модуля вычислений;

– модуля отображения.

Окна данных модулей должны иметь следующее наполнение:

– главное окно (модуль расчетов заключен в нём):

а) три элемента выбора котельной и видов топлива с выпадающим списком выбора котельной и двух видов топлива для сравнения;

б) навигация – кнопки «Отображение результатов» и «Список котельных и видов топлива».

– окно управления данными:

а) две таблицы с отображением списка котельных, нагрузкой на них с их параметрами и отображением списка типов топлива с указанием состава для добавления/изменения/удаления параметров и объектов;

б) кнопки «Обновить», «Удалить».

– окно модуля отображения:

а) гистограмма «Выбросы твердых веществ»;

б) гистограмма «Выбросы двуокиси азота»;

в) гистограмма «Выбросы двуокиси серы»;

г) гистограмма «Удельный и полный объем продуктов сгорания»;

д) таблица, отображающая расчеты в табличном виде с добавлением экономических подсчетов.

Требования к функциям (задачам):

Функциональные требования:

– использование базы данных для хранения данных;

– возможность добавления и изменения пользователем данных параметров котельных и составов топлива, а также самих объектов;

– обработка данных с использованием формул из существующего технологического процесса перевода котельной на альтернативное топливо;

– возможность генерации гистограмм и таблиц, на основе полученных данных.

2 Выбор инструментальных средств и технологий разработки

2.1 Инструментальные средства

Инструментарий разработанного программного комплекса состоит из:

– языка программирования C# с IDE Microsoft Visual Studio 2015;
– системы управления базой данных Microsoft SQL Server;
– инструментария визуализации WPF Toolkit.

Язык программирования и среда разработки

Для начала разработки необходимо было выбрать язык программирования. Существует достаточно широкий спектр языков программирования для создания настольных приложений. Авторы C# стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков (вроде Java) с богатством возможностей и мощью C++. C# позаимствовал большинство своих синтаксических конструкций из C++. В частности, в нем присутствуют такие удобные типы данных, как структуры и перечисления. Синтаксис C# очень похож на синтаксис Java. Например, в C#, также, как и в Java, определение класса состоит из одного файла (*.cs), в отличие от C++, где определение класса разбито на заголовок (*.h) и реализацию (*.cpp).

Как C#, так и Java основаны на синтаксических конструкциях C++. Если Java во многих отношениях можно назвать «очищенной» версией C++, то C# можно охарактеризовать как «очищенную» версию Java.

Синтаксические конструкции C# унаследованы не только от C++, но и от Visual Basic. Например, в C#, как и в Visual Basic, используются свойства классов. Как и в C++, C# позволяет производить перегрузку операторов для

созданных вами типов (Java не поддерживает ни ту, ни другую возможность). Отличием между Java и C# является то, что они созданы для разных типов вычислительных сред.

C# - это фактически гибрид разных языков. При этом C# синтаксически не менее чист, чем Java , прост, как Visual Basic, и обладает практически той же мощью и гибкостью, что и C++ [7].

Основные особенности C#:

– полный и хорошо определенный набор основных типов;

– полная поддержка классов и объектно-ориентированного программирования;

– возможность перегружать операторы, унаследованные от C++. При этом значительная часть возникавших при этом сложностей ликвидирована;

– встроенные синтаксические конструкции для работы с перечислениями, структурами и свойствами классов;

– автоматическое освобождение динамически распределенной памяти;

– возможность отметки классов и методов атрибутами, определяемыми пользователем;

– полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows API;

– возможность использования языка C# для написания WEB-страниц ASP.NET.

Возможно, главной особенностью в языке C# является то, что генерирует код, предназначенный для выполнения только в среде выполнения .NET.

В итоге основными критериями выбора языка для разработки программного комплекса стали:

– имеющиеся навыки работы с данным языком;

– наличие, целесообразность и возможность применения имеющихся библиотек, методов, функций для решения поставленных задач;

– возможность легкой интеграции с СУБД;

– наличие, удобство использования и ширина возможностей IDE (Integrated Development Environment – интегрированная среда разработки).

Так, практические навыки работы из всех языков объективно-ориентированного программирования имелись только с C#, имеющего богатые возможности для создания Desktop-приложений, а разработка велась на платформе .NET, а с учетом существования легких вариантов подключения к БД Microsoft SQL Server с помощью ADO.NET и различных ORM-средств (Entity Framework , NHibernate), наличия инструментария визуализации WPF Toolkit для разработки программного комплекса и был выбран язык объектно-ориентированного программирования C# с IDE Microsoft Visual Studio 2015.

Система управления базой данных

Система управления базами данных (СУБД) — это комплекс программных и языковых средств, предназначенный для управления созданием, ведением и использованием баз данных пользователями.

СУБД включает в себя:

– программные средства создания и поддержания баз данных (стандартная часть);

– сервисные средства (дополнительные возможности).

СУБД в последнее время стали неотъемлемой частью ИТ-инфраструктуры практически любой компании. Если раньше СУБД использовались в основном для хранения текстовых и числовых данных, то сейчас в СУБД хранятся такие данные, как изображения, видеозаписи и многие другие типы данных. Объёмы баз данных в некоторых отраслях выросли до нескольких терабайт.

Согласно рейтингу интернет - журнала Tagline [8], в 2016 году три ведущие позиции по популярности у респондентов, оценивавших СУБД по таким критериям как создание, ведение и использования баз данных занимают:

– MySQL от MySQL AB;

– PostgreSQL, разработанная открытым сообществом;

– SQL Server от Microsoft.

Первые две СУБД являются свободно распространяемыми, Microsoft SQL Server же таковым не является. Рассмотрим подробнее особенности этих СУБД:

MySQL – это свободно распространяемая система управления базами данных с открытым кодом, разработанная компанией MySQL AB. Это высокопроизводительная и масштабируемая СУБД с множеством программных интерфейсов. Она обладает огромными функциональными возможностями и подходит для решения самых разных задач [9].

Преимущества MySQL:

а) MySQL – это кроссплатформенная система и её можно использовать практически во всех современных операционных системах;

б) наличие множества программных интерфейсов, благодаря которым к базе данных MySQL могут подключаться приложения, созданные с помощью C,C++, Java, Perl, PHP, Python, ODBC, NET и Visual Studio;

в) технические характеристики: многопоточность, многопользовательский доступ, быстродействие, масштабируемость;

г) развитая система обеспечения безопасности и разграничения доступа на основе системы привилегий.

К недостаткам MySQL по данным некоторых источников [10] относятся: известные ограничения функционала, проблемы с надежностью из-за некоторых обработки данных, медленная разработка

PostgreSQL — это объектно-реляционная система управления базами данных, основанная на Postgres версии 4.2 - программе, которая была разработана на факультете компьютерных наук Калифорнийского университета в городе Беркли. В Postgres появилось множество новшеств, которые были реализованы в некоторых коммерческих СУБД гораздо позднее Postgres поддерживает большую часть стандарта SQL и предлагает множество современных функций.

Достоинства PostgreSQL:

– свободная распространяемость и открытый исходный код;

– наличие большого сообщества, в котором можно найти ответы на свои вопросы;

– дополнения - несмотря на огромное количество встроенных функций, для PostgreSQL разработано очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими;

– возможность расширения функционала за счет сохранения своих процедур;

– объектность - это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого.

Основными недостатками PostgreSQL являются:

– сложности с установкой. У PostgreSQL нет сертифицированного дистрибутива ни для Linux, ни для Windows;

– отсутствие интеллектуальных подсказок типа IntelliSense при программировании;

– отсутствие полной поддержки самых массовых языков программирования Visual Basic и C#;

– отсутствие примеров кода и литературы на русском;

– некоторые пользователи отмечают проигрыш в скорости относительно MySQL.

Microsoft SQL Server - это система управления реляционными базами данных или RDBMS (Relational Data Base Management System) масштаба предприятия, которая является хорошо масштабируемой, полностью реляционной и быстродействующей. Microsoft SQL Server поддерживает широкий спектр приложений обработки транзакций, бизнес-аналитики и аналитики в корпоративных ИТ-средах. Это одна из трех ведущих в отрасли крупных корпоративных баз данных, наравне с Oracle Database и IBM DB2. Её используют многие ERP и CRM системы, такие как SAP, Microsoft Dynamics, 1С: Предприятие, Microsoft CRM.

Отличительными особенностями Microsoft SQL Server являются:

– шифрование баз данных;

– возможность облачного архивирования и синхронизации в Microsoftt Azure;

– вегкость интеграции с продуктами Microsoft (например, с Visual Studio);

– возможность интеграции с доменом Active Directory. Доступ к данным можно разрешать по доменным пользователям и группам;

– аудит доступа к данным и контроль за осуществляемыми с базой данных действиями;

– наличие служб бизнес-аналитики и анализа данных;

– наличие технологии зеркалирования, обеспечивающей отказоустойчивость СУБД. Данная технология предполагает наличие активного и зеркального серверов с возможностью переключения в случае отказа активного сервера, что происходит либо вручную, либо автоматически посредством использования сервера-наблюдателя;

– возможность «горячего» добавления памяти и процессоров (эта функция должна также поддерживаться и оборудованием);

– репликация БД. В Microsoft SQL Server поддерживаются различные механизмы репликации - с использованием транзакций, слияния и моментальных снимков.

К недостаткам Microsoft SQL Server можно отнести не полную кроссплатформенность – существуют только реализации для UNIX и Windows, при том первая значительно проигрывает второй. Также недостатком Microsoft SQL Server является дороговизна лицензии, особенно в сравнении с MySQL и PostgreSQL, которые распространяются свободно.

Для реализации программного комплекса был выбран именно Microsoft SQL Server ввиду ведения разработки на платформе .NET, для которой в первую очередь он и предназначен, а также использования IDE Microsoft Visual Studio, в которую уже встроен свой локальный сервер и имеются удобные технологии доступа к данным, такие как ADO.NET и различные ORM-технологии, в следствие чего нет необходимости терять время на проблемы с интеграцией.

Инструментарий визуализации

Учитывая требования по генерации гистограмм по результатам выполненных расчетов, и отсутствие встроенных инструментов визуализации, необходимо было определить дополнительный инструментарий.

Им стал WPF Toolkit - коллекция элементов управления WPF, компонентов и утилит номер один для создания приложений Windows на платформе Windows Presentation Foundation. Проект WPF Toolkit загружен более 1 миллиона раз на официальном сайте и на NuGet. Бесплатная версия с открытым исходным кодом Community Edition предоставляет 47 элементов управления и доступна в рамках публичной лицензии Microsoft.

К недостаткам можно отнести отсутствие документации на русском языке.

2.2 Технологии разработки

В разработанном программном комплексе задействуется две технологии:

– платформа Windows Presentation Foundation (WPF) с языком разметки XAML;

– ORM - технология Entity Framework 6.

Windows Presentation Foundation

Согласно требованиям, задачей являлось разработка настольного приложения. Язык C# предлагает несколько вариантов интерфейсов программирования приложений (Application Programming Interface - API) для разработки настольных приложений, наиболее популярными из них являются Windows Presentation Foundation (WPF) и Windows Forms.

Обе эта этих интерфейса предлагают схожий функционал, но Windows Forms появился еще в .NET Framework 1.0, а с выходом .NET Framework 3.0 Microsoft выпустила Windows Presentation Foundation, который базировался на DirectX 11 и декларативном языке описания интерфейсов XAML. WPF является «улучшенной» версия Windows Forms, предлагая дополнительные возможности. Компания-разработчик (Microsoft), однако, не отказывается от Windows Forms, но еще в 2014 представители компании заявили, что новые функции в данный API добавляться не будут.

WPF (Windows Presentation Foundation) является частью экосистемы платформы .NET и представляет собой подсистему построения пользовательских интерфейсов для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения,

ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Также платформа Windows Presentation Foundation предоставляет инфраструктуру разработки, предназначенную для построения высококачественных пользовательских интерфейсов для операционной системы Windows [11]. WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, документы и безопасность.

Также WPF использует механизм привязок данных (bindings), который обеспечивает простой и последовательный способ, обеспечивающий взаимодействие приложения с данными. Можно связывать элементы с данными из разнообразных источников данных в форме объектов среды CLR и XML. Привязка данных является процессом, который устанавливает связь между пользовательским интерфейсом приложения и его бизнес-логикой. Если привязка имеет правильные параметры и данные предоставляют правильные уведомления, то при изменении значений данных в элементах, которые привязаны к данным, автоматически отражаются изменения. Привязка к данным может также означать, что, если внешнее представление данных в элементе изменяется, то базовые данные могут автоматически обновляться для отражения изменений.

WPF использует расширяемый язык разметки для приложений (XAML), чтобы предоставить декларативную модель для программирования приложений.

XAML (eXtensible Application Markup Language) - язык разметки, используемый для инициализации объектов в технологиях на платформе .NET. С точки зрения модели программирования .NET Framework язык XAML упрощает создание пользовательского интерфейса. Можно создать видимые элементы пользовательского интерфейса в декларативной разметке XAML, а затем отделить определение пользовательского интерфейса от логики времени выполнения, используя файлы кода программной части, присоединенные к разметке с помощью определений разделяемых классов.

Язык XAML напрямую представляет создание экземпляров объектов в конкретном наборе резервных типов, определенных в сборках. В этом заключается его отличие от большинства других языков разметки, которые, как правило, представляют собой интерпретируемые языки без прямой связи с системой резервных типов. Язык XAML обеспечивает рабочий процесс, позволяющий нескольким участникам разрабатывать пользовательский интерфейс и логику приложения, используя потенциально различные средства.

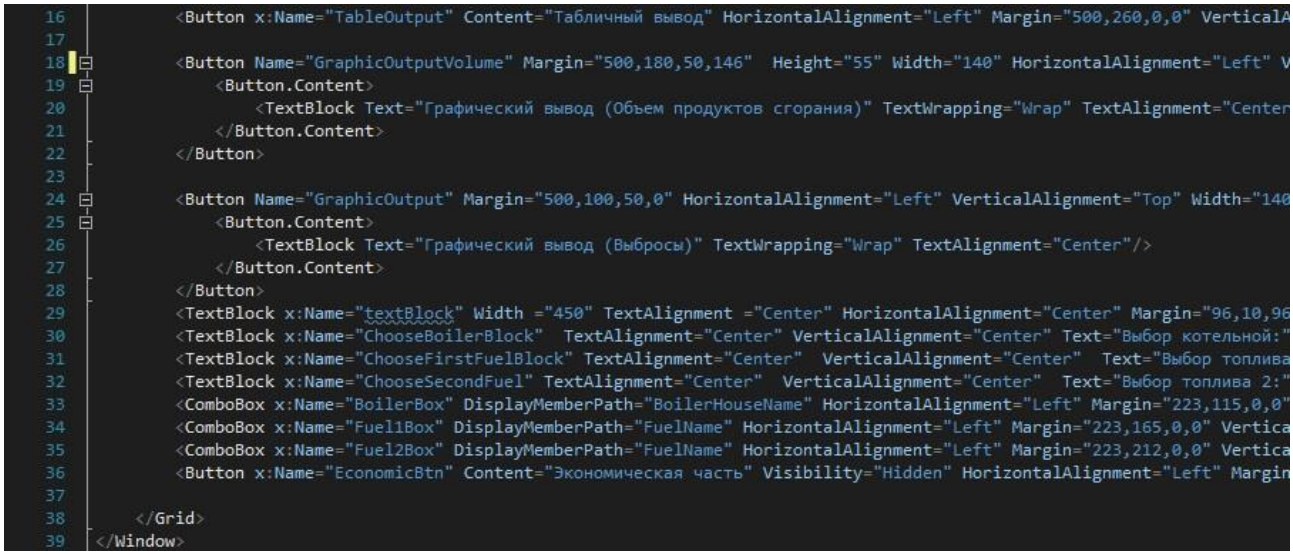


Рисунок 7 - Пример XAML - разметки

При представлении в виде текста файлы XAML являются XML-файлами, которые обычно имеют расширение .xaml. Файлы можно сохранять в любой кодировке, поддерживаемой XML, но обычно используется кодировка UTF-8.

Также WPF представляет собой обширный API-интерфейс для создания настольных графических программ имеющих насыщенный дизайн и интерактивность. В отличие от устаревшей технологии Windows Forms, WPF включает новую модель построения пользовательских приложений (в основе WPF лежит мощная инфраструктура, основанная на DirectX).

Это означает возможность применения развитых графических эффектов, не платя за это производительностью, как это было в Windows Forms. Фактически даже становятся доступными такие расширенные средства, как поддержка видеофайлов и трехмерное содержимое. Используя эти средства можно создавать пользовательские интерфейсы и визуальные эффекты, которые были просто невозможны в Windows Forms.

Преимущества WPF:

– возможность определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#;

– независимость от разрешения экрана: поскольку в WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением;

– новые возможности, которых сложно было достичь в WinForms, например, создание трехмерных моделей, привязка данных, использование таких элементов, как стили, шаблоны, темы и других;

– хорошее взаимодействие с WinForms, благодаря чему, например, в приложениях WPF можно использовать традиционные элементы управления из WinForms;

– богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самостоятельно создавать новые элементы;

– аппаратное ускорение графики - вне зависимости от того, работаете ли вы с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной;

– создание приложений под множество ОС семейства Windows - от Windows XP до Windows 10.

В тоже время WPF имеет определенные ограничения. Несмотря на поддержку трехмерной визуализации, для создания приложений с большим количеством трехмерных изображений, прежде всего игр, лучше использовать другие средства - DirectX или специальные фреймворки, такие как Monogame или Unity.

Также стоит учитывать, что по сравнению с приложениями на Windows Forms объем программ на WPF и потребление ими памяти в процессе работы в среднем несколько выше. Но это с лихвой компенсируется более широкими графическими возможностями и повышенной производительностью при отрисовке графики [12].

Для разработки программного комплекса была выбран API WPF по причине того, что она является наиболее современной для разработки приложений на C#, а также ввиду наличия в ней проработанной графической составляющей, что важно для реализации визуализации данных, весомым преимуществом WPF является также то, поддерживает механизм «связывания» данных, который позволяет указать привязку широкого диапазона свойств к различным источникам данных, а также реализовать различные архитектурные паттерны.

ORM Entity Framework

Для получения доступа к данным (в данном случае, к БД MS SQL) на платформе .NET существует несколько вариантов, первый – использовать традиционные средства ADO.NET (ActiveX Data Object для .NET), но на данный момент существуют различные технологии, базирующиеся на ADO.NET и позволяющие еще больше упростить доступ к данным и сделать его более удобным. Одним из них является Entity Framework – объектно-ориентированная (ORM – Object Relational Mapping) технология доступа к

данным, представляющая собой набор технологий ADO.NET, обеспечивающих разработку приложений, связанных с обработкой данных. Entity Framework позволяет разработчикам создавать приложения для доступа к данным и работы с ними, работающие с концептуальной моделью приложения, а не напрямую с реляционной схемой хранения. Цель этого состоит в уменьшении итогового объема кода и снижении затрат на сопровождение приложений, ориентированных на обработку данных [13].

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет работать с данными вне зависимости от типа хранилища. Если на физическом уровне мы оперируем таблицами, атрибутами, первичными и внешними ключами, то на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Центральной концепцией Entity Framework является понятие сущности – entity. Сущность или Entity - это набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, для сущности, описывающей автомобиль можно выделить такие свойства, как марка, модель, длина, ширина, возраст, масса, мощность двигателя. Свойства необязательно представляют простые данные типа int, string но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально её определять. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи. Архитектура Entity Framework представлена на рисунке 7.

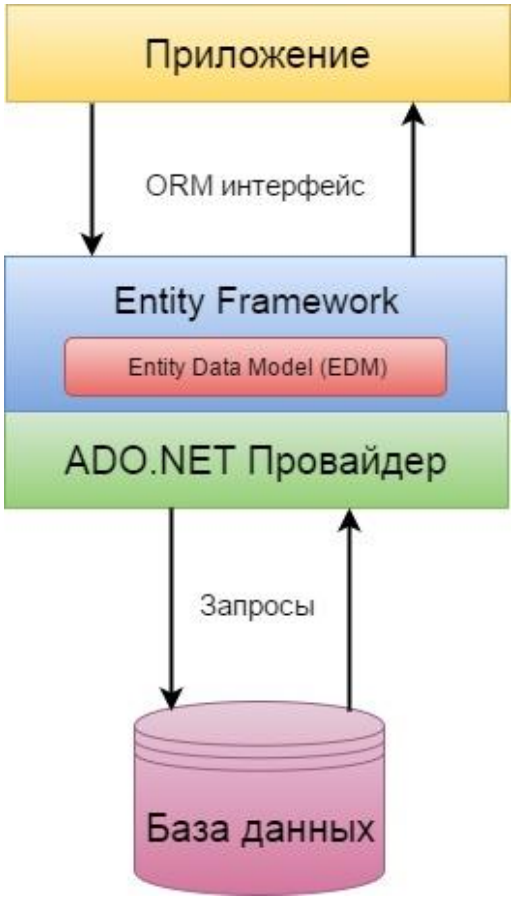


Рисунок 8 - Архитектура Entity Framework

Отличительной чертой Entity Framework является использование запросов на языке LINQ - Language Integrated Query для выборки данных из базы. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из базы данных, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней:

– концептуального;
– уровня хранилища;
– уровня сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, и отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Рисунок 9 – Схема Entity Data Model

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Способы взаимодействия Entity Framework с базой данных:

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

– Database First - Entity Framework создаёт набор классов, которые отражают модель существующей базы данных;

– Model First - разработчик создает модель базы данных (.edmx), по которой затем Entity Framework создает реальную базу данных на сервере;

– Code First - разработчик создает код модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели создает базу данных и ее таблицы.

Изначально с самой первой версии Entity Framework поддерживал только Database First, Позже был добавлен подход Model First. Начиная с 5.0 предпочтительным подходом становится Code First.

Схема способов взаимодействия представлена на рисунке 10

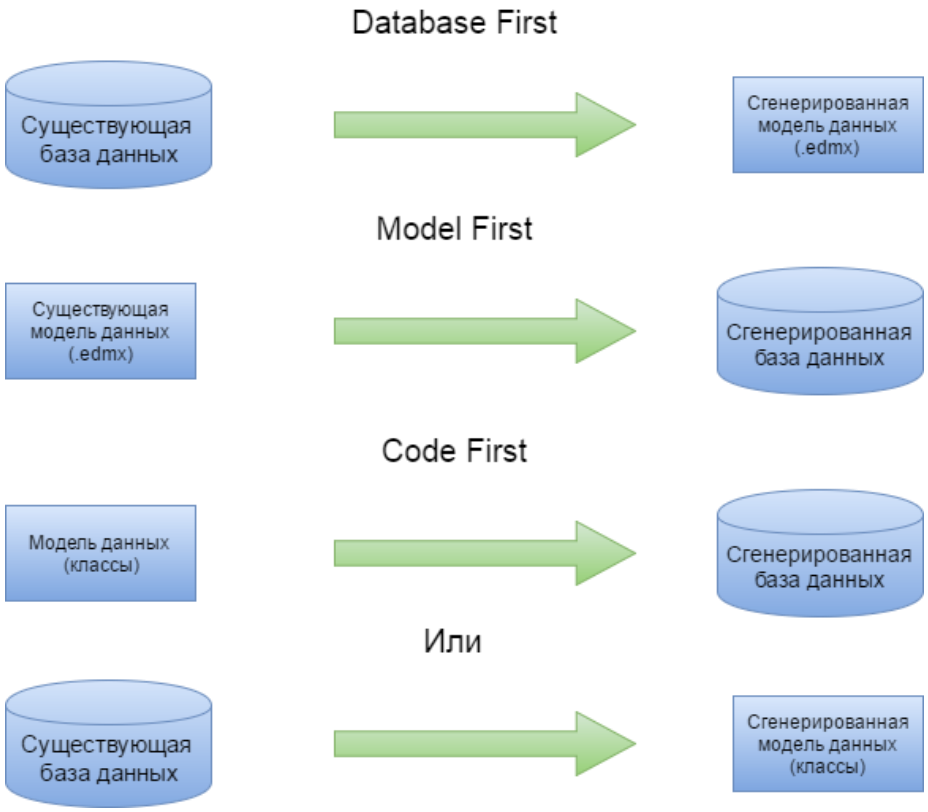


Рисунок 10 – Способы взаимодействия Entity Framework с базой данных

Основными возможности Entity Framework являются:

– поддержка работы с различными серверами баз данных (Microsoft SQL Server, Oracle, DB2);

– возможность строить модель по схемам физических баз данных, а также хранимым процедурам;

– интеграцией со IDE Visual Studio для визуальной и автоматической генерации модели по имеющимся базам данных;

– возможность сгенерировать модель и базу данных на ее основе по семантическому описанию (подход Code First).

Как показывает опыт, времени на проектирование и написание кода, как правило, всегда мало – и архитекторы, и разработчики ищут пути наименьшего сопротивления. В большинстве ситуаций технология ADO.Net себя не оправдывает, и её применение в некоторых проектах обусловлено совместимостью с ранее написанным кодом. Однако в силу достаточного большого возраста технологии, у разработчиков накоплено значительное

количество наработок – фреймворков, оберток, вспомогательных классов, применение которых несколько сглаживает недостатки ADO.Net. В новых же проектах всегда используются технологии Entity Framework или LINQ to SQL, в которых возможность автоматического создания классов из сущностей базы данных значительно уменьшает время разработки, сокращает размер написанного кода, делая его более понятным.

Главные преимущества, которые дает использование Entity Framework:

– сокращение времени разработки, возможность уделять больше времени логике приложения, а не способам взаимодействия с данными;

– взаимодействие с данными осуществляется по концептуальной модели, а не конкретной структуре хранения самих данных, что в свою очередь позволяет разрабатывать базовые компоненты, вне зависимости от данных, с которыми они работают;

– позволяет избежать жестко прописанных зависимостей обработки данных в коде приложения;

– изменения в схеме данных автоматически применяются к объектно-реляционной модели этих данных, генерируемой платформой Entity Framework;

– гибкая поддержка языка интегрированных запросов (Language-Integrated Query, LINQ), что позволяет автоматизировать операции группировки, сортировки и фильтрации данных.

Для разработки программного комплекса Entity Framework был выбран в виду легкой интеграции с Visual Studio (его можно за несколько секунд установить с помощью NuGet), а также возможности применения подхода Code-First, который для меня, как для разработчика является наиболее предпочтительным.

3 Разработка программного комплекса

3.1 Структура программного комплекса

Технологии разработки настольных приложений на платформе .NET, такие как Windows Forms, WPF, Silverlight предоставляют опыт, который ведет разработчика по пути перетаскивания элементов управления из панели инструментов на поверхность дизайна, а затем записаывать код в файл кода, находящийся на форме. Поскольку такие приложения растут по размеру и масштабу и изменяются, возникают сложности с обслуживанием. Эти проблемы включают в себя тесную связь между элементами управления пользовательского интерфейса и бизнес-логикой, что увеличивает затраты на модификацию пользовательского интерфейса и сложность модульного тестирования такого кода.

С учетом вышесказанного, целесообразным является использование такой структуры приложениях, которая позволила бы избежать этих проблем. И такое решение существует. Для организации структуры программного комплекса было решено использовать архитектурный шаблон Model-View-ViewModel (MVVM).

MVVM - это архитектурный шаблон (паттерн), он задает общую архитектуру приложения. Его цель - обеспечить четкое разделение проблем между элементами управления пользовательским интерфейсом и их логикой.

Основными мотивами реализации приложения с использованием шаблона MVVM являются:

– разделение властей. Тесно связанный, устойчивый к изменениям, хрупкий код вызывает все виды долгосрочных проблем обслуживания, которые в конечном итоге приводят к плохой удовлетворенности клиентов поставляемым программным обеспечением. Чистое разделение между логикой приложения и пользовательским интерфейсом упростит тестирование, поддержку и развитие приложения.

– родственность. MVVM - естественная модель для платформ XAML. Ключевыми элементами шаблона MVVM являются богатый стек привязки данных и свойства зависимостей. Комбинация этих средств обеспечивает возможность подключения пользовательского интерфейса к модели представления.

– возможность работать с разработчиком-дизайнером. Когда пользовательский интерфейс XAML не тесно связан с кодовым отставанием, разработчикам легко реализовать свободу, необходимую им для творчества и сделать хороший продукт.

– повышение возможности тестирования приложений. Перемещение логики пользовательского интерфейса в отдельный класс, который может быть создан независимо от технологии пользовательского интерфейса, значительно упрощает модульное тестирование.

В шаблоне MVVM есть три основных компонента: модель, представление и модель представления. Каждая из них служит отдельной и отдельной роли. На следующем рисунке показаны взаимосвязи между тремя компонентами.

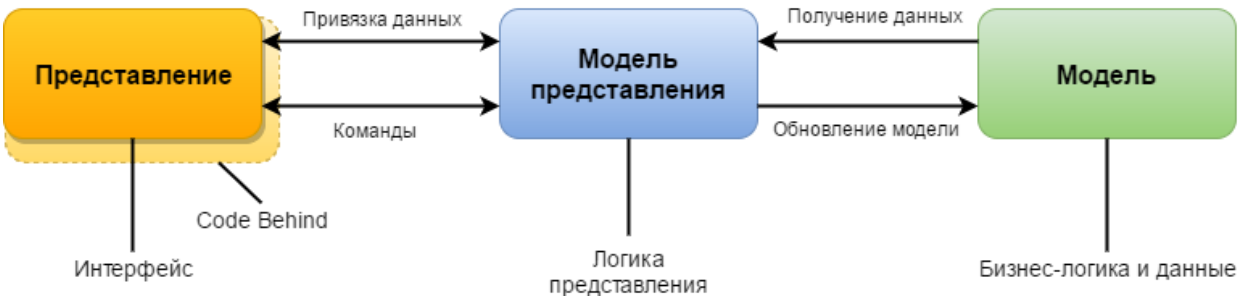


Рисунок 11 – Схема шаблона Model-View-ViewModel (MVVM)

Модель (Model) описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику проверки свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

Представление (View) отвечает за определение структуры, макета и внешнего вида того, что пользователь видит на экране. В идеальном случае представление определяется исключительно с помощью XAML с ограниченным кодом, который не содержит бизнес-логики. Хотя окно (класс Window) в WPF может содержать как интерфейс в XAML, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики. Вся же основная логика приложения выносится в компонент ViewModel. Представление содержит лишь ссылки на модель и на модель представления.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel. Например, представление может содержать код в файле кода, который приводит к тому, что модель представления назначается как свойство DataContext.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

Модель представления (ViewModel) выступает в качестве посредника между представлением и моделью и отвечает за обработку логики представления и связывает модель и представление через механизм привязки данных (Bindings). Модель представления извлекает данные из модели, а затем делает данные доступными для представления и может каким-то образом переформатировать данные, что упрощает обработку представления. Модель представления также обеспечивает реализацию команд, которые пользователь приложения инициирует в представлении. Например, когда пользователь нажимает кнопку в пользовательском интерфейсе, это действие может вызвать команду в модели представления.

Итогом применения шаблона MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать. Структура созданного проекта отображена на рисунке 12.

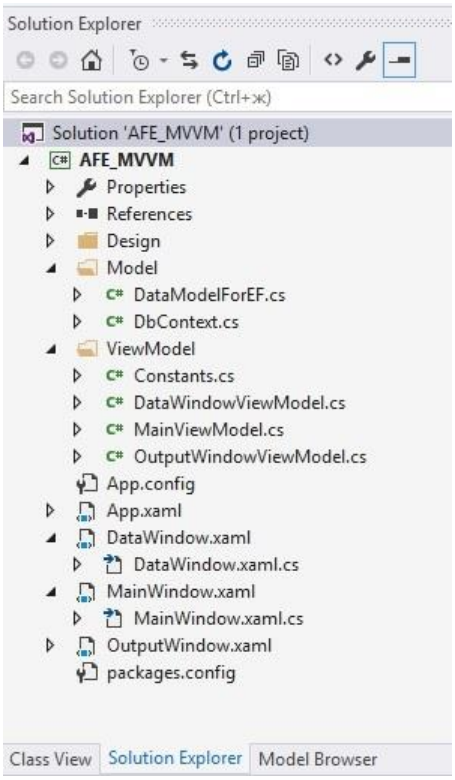


Рисунок 12 – Структура проекта

Каждому представлению, организованному в виде .xaml - окна соответствует своя модель представления, расположенная в папке ViewModel. Для выполнения действий в представлении, не используются обработчики кнопок, используются команды-объекты класса RelayCommand, тело которого представлено на рисунке 13.

```
public class RelayCommand : ICommand
{
    public RelayCommand(Action execute);
    public RelayCommand(Action execute, Func<bool> canExecute);

    public event EventHandler CanExecuteChanged;

    public bool CanExecute(object parameter);
    public virtual void Execute(object parameter);
    public void RaiseCanExecuteChanged();
}
```

Рисунок 13 - Тело класса RelayCommand

На рисунке 14 отображен пример использования команды в XAML – разметке.

```
<Button Command ="{Binding ShowBoilersCommand}" Grid.Row="0" Height="25" Width="60"
        Margin="350,0,0,0">Котельные</Button>
<Button Command ="{Binding ShowFuelsCommand}"  Grid.Row="0" Height="25" Width="70"
        Margin="200,0,0,0">Топливо</Button>
```

Рисунок 14 – Пример использования команд в XAML – разметке окна «DataWindow»

3.2 Реализация модулей программного комплекса

Реализация модуля управления данными

Для организации доступа к данным применен подход Code-First для Entity Framework, схема которого отображена на рисунке 15. Необходимо было создать код модели данных (определить классы и заполнить их поля), которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

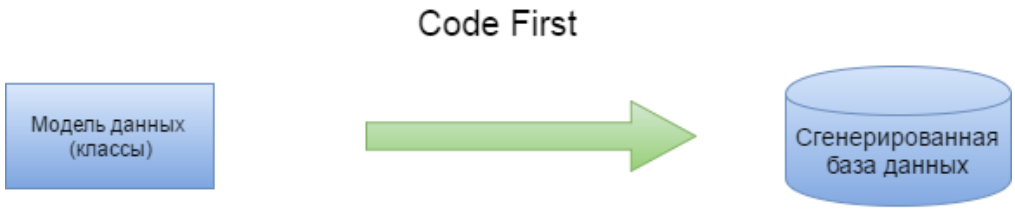


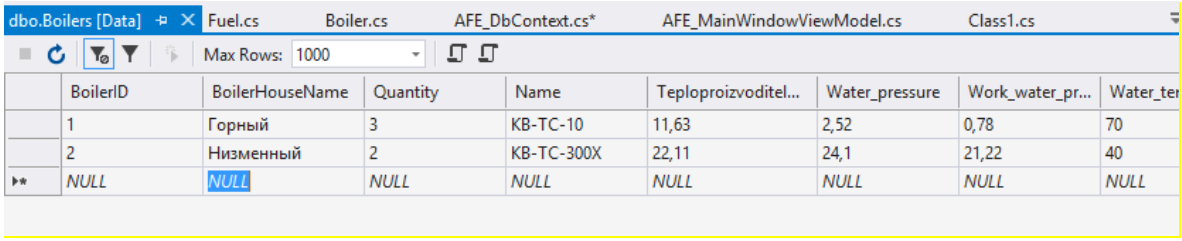
Рисунок 15 – Подход Code-First

Согласно данному подходу были созданы основные классы, формирующие таблицы БД: Fuel и Boilers. Так как проект перерабатывался несколько раз, с учетом существующей и заполненной БД было разумно использовать подход Code Second, при применении которого также создаются классы, имена и тип полей которых совпадает с атрибутами в таблицах БД.

```
namespace Model
{
    public class Fuel
    {
        public int Fuel_ID { get; set; }
        public string FuelName { get; set; }
        public double Wp { get; set; }
        public double Ap { get; set; }
        public double Sk { get; set; }
        public double Sop { get; set; }
        public double Cp { get; set; }
        public double Hp { get; set; }
        public double Np { get; set; }
        public double Qp { get; set; }
        public double Qn { get; set; }
    }
}
```

Рисунок 16 - Тело класса «Fuel»

На рисунке 16 отображено тело класса Fuel с полями, которые в итоге соответствуют столбцам одноименной таблицы базы данных. Вид заполненной таблицы в Microsoft SQL Server отображен на рисунке 17.



BoilerID	BoilerHouseName	Quantity	Name	Teploproizvoditel...	Water_pressure	Work_water_pr...	Water_ter
1	Горный	3	КВ-ТС-10	11,63	2,52	0,78	70
2	Низменный	2	КВ-ТС-300Х	22,11	24,1	21,22	40
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 17 – Таблицы «Fuel» в Microsoft SQL Server

Реализация модуля расчетов

Алгоритм осуществления расчетов заключается в осуществлении запросов к базе данных по выбранным в главном окне в элементах управления ComboBox видов топлива и котельной, запрашиваются два объекта вида топлива, и один объект котельной, а также используется класс констант Constants. В дальнейшем используются поля этих объектов и константы в формулах, которые реализуются методами CalcMso2, CalcMtv, CalcMno2, Calcuvg.

Реализация модуля отображения

Для отображения результатов расчетов в графическом виде используются элементы управления из WPF Toolkit. В данном случае используется элемент Chart с типом ColumnSeries. Генерация графиков происходит на основе результатов расчетов, записанных в коллекции, а наименования столбцов согласно выбранным данным в выпадающих списках на главном окне.

Для реализации табличного вывода используется элемент DataGrid, который также заполняется из коллекции.

3.3 Описание пользовательского интерфейса

На главном окне приложения, изображенном на рисунке 18, имеются выпадающие списки выбора видов топлива и выбора котельной, а также переходы с помощью кнопок к окну модуля управления данными и к окну модуля отображения.

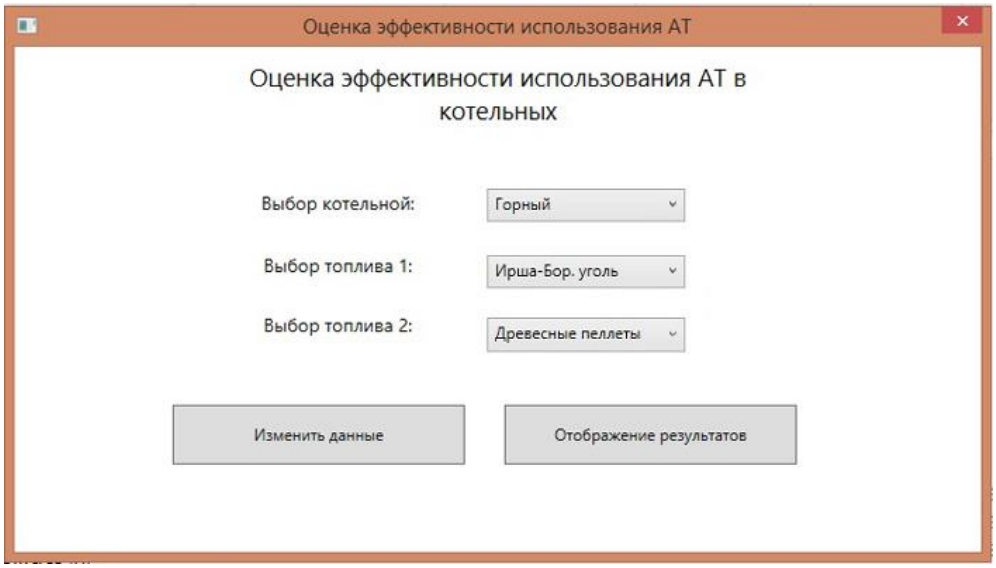


Рисунок 18 – Главное окно программного комплекса

На рисунке 19 отображено окно управления данными о котельных и видах топлива, в котором имеются две таблицы с данными, поддерживающие редактирование, и кнопки для совершения действий.

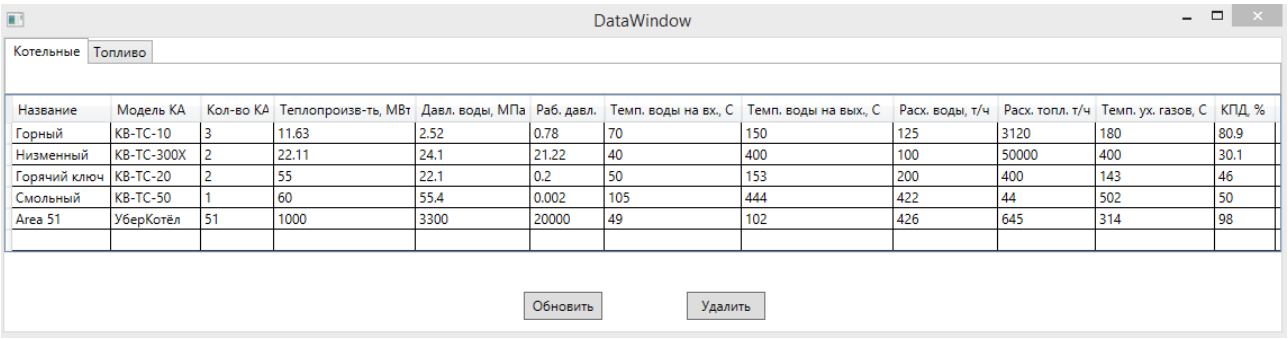


Рисунок 19 – Окно модуля управления данными

На рисунке 20 отображено окно модуля отображение со вкладками: «Выбросы графически», в которой отображаются гистограммы по выбросам золы, двуокиси серы и азота, «Объем продуктов сгорания», на которой отображаются гистограммы по полному и удельному объему продуктов сгорания и «Табличный вывод», на котором вся информация собрана в таблице. На рисунке 20 активна вкладка «Выбросы графически».

Рисунок 20 – Окно графического отображения расчётного количества выбросов

ЗАКЛЮЧЕНИЕ

В ходе выполнения бакалаврской работы был проведен анализ предметной области и изучен технологический процесс перевода котельной на альтернативное топливо. С целью автоматизации инженерных расчетов данного техпроцесса и было принято решение о разработке программного комплекса. Для проектируемого комплекса были построены схема бизнес процесса и диаграмма прецедентов, определены необходимые требования, программные средства и соответствующие технологии разработки. Мною были освоены и приобретены навыки программирования на языке C# с использованием платформы WPF и при помощи паттерна архитектурного проектирования MVVM. Были получены навыки управления реляционными базами данных и в дальнейшем использованы для организации данных с помощью MS SQL для удобства доступа к данным была освоена ORM – технология Entity Framework 6.

Данная разработка была апробирована на заочных конференциях:

– «Современные тенденции развития науки и производства», г. Кемерово, 27-28 октября 2016 г.

– «Современные тенденции развития науки и технологий», г. Белгород, 31 января 2017 г.

– «Актуальные вопросы технических наук. IV Международная научная конференция», г. Краснодар, февраль 2017;

В журналах:

– «Научный форум. Сибирь» г.Тюмень. №4, 2016. Том 2, с. 77-79;

– «Евразийский союз ученых» г.Москва, №31, 2016. Часть, с.78-81;

– «Молодой ученый» г.Казань. №23, 2016. Часть 1, с. 81-84.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. International Energy Outlook 2016 [Электронный ресурс] : научный журнал // U.S. Energy Information Administration. - Режим доступа: http://www.eia.gov/outlooks/ieo/pdf/0484(2016).pdf.
2. Истягина Е.Б. Технологический процесс перевода котельной на альтернативное топливо / Е. Б. Истягина, С. Е. Молоков // Образовательные ресурсы и технологии. – 2016. – №2. – 5 с.
3. Лазарева О. Н. Технико-экономические показатели проектируемой котельной : метод. указ. по дипломному проектированию для студентов спец.1007 / О. Н. Лазарева, Г. Б. Максумов ; Краснояр. гос. техн. ун-т. – 1997
4. Автоматизация инженерных расчётов : методические указания по выполнению курсовой работы / сост. : Г.В. Мозгова, М.Ю. Серёгин, И.П. Борисов, П.В. Балабанов. – Тамбов : Тамб. гос. техн. ун-т, 2010. – 40 с.
5. Буч, Г. Язык UML. Руководствао пользователя / Г. Буч, Д. Рамбо, А. Якобсон. – Москва: ДМК Пресс, 2007. – 496 с
6. ГОСТ 34.602-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Введ. 01.01.1990. – Москва: Стандартинформ, 2018. – 16 с.
7. Троеслен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – Москва : Вильямс, 2015 – 1312 с.
8. Рейтинг систем управления базами данных 2016 [Электронный ресурс] // Tagline – рейтинги сервисов и технологий - Режим доступа: http://tagline.ru/database-management-systems-rating.
9. Гольцман, В. MySQL 5.0. Библиотека программиста / В. Гольцман. – Санкт-Петербург : Питер, 2010. - 431 с.

10. SQLite vs MySQL vs PostgreSQL [Электронный ресурс] // DevAcademy – обучение современному программированию – Режим доступа: http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql

11. Макдоналд, М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов / М. Макдоналд. – Москва: Вильямс, 2013. – 1045 с.

12. Паттерн MVVM, определение паттерна MVVM [Электронный ресурс] // Metanit – сайт о программированиии. – Режим доступа: https://metanit.com/sharp/wpf/22.1.php

13. Общие сведения о платформе Entity Framework [Электронный ресурс] // Microsoft Developer Network – Режим доступа: https://msdn.microsoft.com/ru-ru/library/bb399567(v=vs.110).aspx

14. СТО 4.2–07–2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной и научной деятельности. - Введ. 30.12.2013. - Красноярск : ИПК СФУ, 2014. - 60 с.

ПРИЛОЖЕНИЕ А

Инженерные расчеты из технологического процесса

Расчет расхода топлива, объема теоретического количества воздуха и продуктов сгорания, при сжигании бурого угля:

Исходные данные:

Топливо: Ирша-Бородинский уголь.

Состав топлива:

– Wp= 33%;

– Ap = 6,0%;

– Sк = 0,2%;

– Sop = 0,2%;

– Cp = 43,7%;

– Hp = 3%;

– Np =4,6%;

– Op=13,5.

Низшая теплота сгорания: $Q_{н}$= 15,67 МДж/кг.

Для всей Котельной находим расход топлива, (г/с):

$$B^{p} = \frac{N \cdot Q_{вк}}{\eta_{ка} \cdot Q_{н}^{p}}, \quad (1)$$

где $Q_{н}^{р}$ – низшая теплота сгорания 1 кг твердого топлива,МДж/кг;

N – число котлоагрегатов, шт;

$\eta_{ка}$ – коэффициент полезного действия котлоагрегата, брутто;

$Q_{ВК}$ – полное количество полезно использованного тепла, МДж.

Определяем теоретически необходимое количество воздуха для сгорания 1 кг твердого топлива, м3/кг:

$$V_o = \frac{V_{o_2}^o}{0,21} = 0,89\left(C^p + 0,375S_{op+к}^p\right) + 0,265H^p - 0,0333O^p, \quad (2)$$

Объем трехатомных сухих газов в сумме с теоретическим объемом азота и водяного пара, $м^3/м^3$:

$$V_o^Г = V_{RO_2} + V_{N_2}^o + V_{H_2O}^o, \quad (3)$$

$$V_{RO_2} = V_{CO_2} + V_{SO_2} = \frac{1,866}{100}\left(C^p + 0,375S_{op+к}^p\right), \quad (4)$$

$$V_{N_2}^o = 0,79 \cdot V_o + 0,8 \cdot \frac{N^p}{100}, \quad (5)$$

$$V_{H_2O}^o = 0,111 \cdot H^p + 0,0124 \cdot W^p + 0,0161 \cdot V^o, \quad (6)$$

Температурная поправка на увеличения объема дымовых газов при условиях отличных от нормальных:

$$\theta = \frac{t_{УХГ} + 273}{273}, \quad (7)$$

где $t_{УХ.Г}$ – температура уходящих газов, °С.

Определим коэффициент избытка воздуха на выходе из дымовой трубы:

$$\alpha_{ТР} = \alpha_{Т} + \alpha_1 + \alpha_2, \quad (8)$$

где $\alpha_{\text{тр}}$ – коэффициент избытка воздуха в топочной камере;

α_1, α_2 – коэффициент избытка воздуха по газоходам;

Полный объем продуктов сгорания с учетом коэффициента избытка воздуха, м^3/кг:

$$V_{Г} = B^{р} \cdot \left[V_{о}^{Г} + 1{,}016 \cdot V_{о} (\alpha_{ТР} - 1) \right] \cdot \theta, \quad (9)$$

Удельный объем продуктов сгорания с учетом коэффициента избытка воздуха, м^3/кг:

$$V_{Г}^{УД} = V_{о}^{Г} + (\alpha_{ТР} - 1) \cdot V_{о}, \quad (10)$$

Расчет выбросов вредных веществ

Определяем массовый выброс летучей золы, г/с:

$$M_{ТВ} = 0{,}01 B^{р} \cdot \alpha_{УН} \left(A^{р} + q_4 \frac{Q_{н}^{р}}{32{,}680} \right) \cdot (1 - \eta_{з}), \quad (11)$$

где $\alpha_{\text{УН}}$ – доля золы топлива уносимая газами;

η_3 – доля твердых частиц улавливаемых в золоуловителе;

q_4 – потери теплоты от механической неполноты сгорания топлива.

Концентрация выбросов частиц золы, г/м^3:

$$C_{TB} = \frac{M_{TB}}{V_Г}, \quad (12)$$

Массовый выброс оксидов серы SO_2 и SO_3 в пересчете на SO_2

Определяем количество оксидов серы в пересчете на SO_2, г/с:

$$M_{SO_2} = 0,02B \cdot S^p \cdot \left(1 - \eta'_{SO_2}\right) \cdot \left(1 - \eta''_{SO_2}\right), \quad (13)$$

где η'_{SO_2} – доля оксидов серы, связываемых в газовом тракте котла за счет реакций протекающих в минеральной части топлив;

η''_{SO_2} – доля оксидов серы, улавливаемых в золоуловителе. Она является функцией приведенной сернитости топлива.

Концентрация выбросов оксидов серы, г/м^3:

$$C_{SO_2} = \frac{M_{SO_2}}{V_Г}, \quad (14)$$

Массовый выброс окислов азота в пересчете на NO_2:

Определяем коэффициент K, характеризующий выход оксидов азота:

$$K = \frac{25 \cdot Q_ф}{200 + Q}, \quad (15)$$

где $Q_ф$ и Q – фактическая и номинальная тепловая мощность котла соответственно, МВт.

Определяем количество окислов азота в пересчете на NO2 выбрасываемых в атмосферу с дымовыми газами котельных установок, г/с:

$$M_{NO_2} = 0{,}34 \cdot 10^{-7} \cdot K \cdot B \cdot Q^p \cdot \left(1 - \frac{Q_4}{100}\right) \cdot (1 - \varepsilon_1 \cdot r) \cdot \beta_1 \cdot \beta_2 \cdot \beta_3 \cdot \varepsilon_2, \quad (16)$$

где ε_1 – коэффициент, характеризующий эффективность рециркуляции газов в зависимости от условий подачи их в топку,%;

ε_2 – коэффициент, характеризующий снижение выбросов оксидов азота,%;

r – степень рециркуляции дымовых газов,%;

β_1 - коэффициент, учитывающий влияние на выход оксидов азота в качестве сжигаемого топлива,%;

β_2 - коэффициент, учитывающий конструкцию горелок,%;

β_3 - коэффициент, учитывающий вид шлакоудаления,%;

Концентрация выбросов оксидов азота, г/м^3:

$$C_{NO_2} = \frac{M_{NO_2}}{V_Г}, \quad (17)$$